PHIL 7001: Fundamentals of AI, Data, and Algorithms

Week 8
Logistic Regression

Boris Babic,
HKU 100 Associate Professor of Data Science, Law and Philosophy

Today

Learning goals

- Introduce the fundamental concepts of logistic regression.
- Understand the sigmoid function and its role in logistic regression.
- Implement logistic regression for binary classification using real-world examples.
- Evaluate the performance of a logistic regression classifier using appropriate metrics.
- Interpret and analyze ROC curves and AUC values.

Overview

- Before we delve into the technicalities, let's grasp the essence of logistic regression.
- Logistic regression model (also known as logit model) is often used for classification and predictive analytics
- The outcome of the logit model must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc.
- However, instead of giving the exact value as 0 and 1, it gives the probability of the instance being that class.
  - i.e. learn $p(y|x)$
- Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

- Recall that the equation of the best-fit line in linear regression is:

$$y = \beta_0 + \beta_1 x$$

  where x is the predictor, β_0 and β_1 are the model coefficients
- Let's say instead of y we are taking probabilities (p). But there is an issue here, the value of (p) will exceed 1 or go below 0, and we know that the range of a probability must be between 0 and 1.
- To overcome this issue we take "odds" of p
- The odds in favor of an event is the quantity $\frac{p}{1-p}$, where p is the probability of the event (i.e. probability that y=1).
  - $\frac{p}{1-p} = \beta_0 + \beta_1 x$

The Logit Link

- In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.
- This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 x$$

- In other words, logistic regression assumes that the log odds is a linear function of x.
- We can also write the function as

$$logit(pi) = \beta_0 + \beta_1 x$$

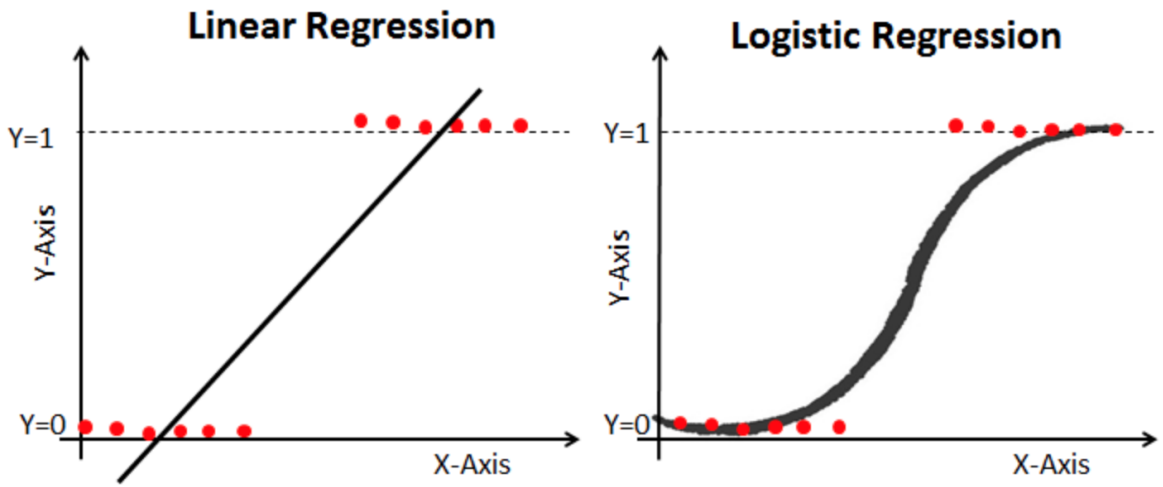
where logit(pi) is the log-odds (logarithm of odds) of the positive outcome.

Sigmoid Function

- Since we want to predict probability rather than getting the 'log of odds', we can derive the function of p by taking the exponent on both sides and then solving for p.

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- Now we have our logistic function, also called a sigmoid function.
- The sigmoid function maps the predicted values to probabilities.
- Note that the function always lies in the range of 0 to 1, boundaries being asymptotic, so it forms a curve like the "S" form.



Linear regression vs Logistic regression

- Linear Regression
  - Linear regression models are used to identify the relationship between a continuous dependent variable and one or more independent variables.
  - For each type of linear regression, it seeks to plot a line of best fit through a set of data points, which is typically calculated using the least squares method.
- Logistic Regression
  - Logistic regression is used to make a prediction about a categorical variable versus a continuous one. A categorical variable can be true or false, yes or no, 1 or 0, etc.
  - The unit of measure also differs from linear regression as it produces a probability, but the logit function transforms the S-curve into straight line.

Example

- Imagine we are dealing with a dataset of students and their study hours, and we want to predict whether a student will pass or fail an exam based on their study hours.
- We collect data on students' study hours and whether they passed or failed the exam.
- We have two variables: study_hours (predictor) and exam_result (response: pass/fail).

```
# Display the first few rows of the data
glimpse(data)
```

```
## Rows: 100
## Columns: 2
## $ study_hours <dbl> 2.9, 7.9, 4.1, 8.8, 9.4, 0.5, 5.3, 8.9, 5.5, 4.6, 9.6, 4.5…
## $ exam_result <chr> "Fail", "Pass", "Pass", "Fail", "Pass", "Fail", "Fail", "P…
```

Example - Logistic Regression Model

- Since our response variable of interest is whether the student passed the exam, Y is a binary variable that can be either 1 or 0.
- Our model equation using the logistic link function becomes:

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 study\_hours$$

- Here, p is the probability of passing the exam given a student's study hours.

Example - Logistic Regression Model

- Before building the model, let's first transform the predictor study_hours into binary (0 or 1). The variable will take the value 1 if the student passed the exam, and 0 otherwise.

```
# Transform pass/fail outcomes into binary (0 or 1)
data$exam_result <- ifelse(data$exam_result == "Pass", 1, 0)
```

```
# Logistic Regression Model
model <- glm(exam_result ~ study_hours, data = data, family = binomial(link = "logit"))
```

- In R, we can build the logistic regression model via the **glm()** function.
- The function component **'family = binomial(link = "logit")'** is telling R we want to use the logit link function.

Example - Logistic Regression Model

```
summary(model)$coefficients
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -3.3520313  0.7132017 -4.699977 2.601911e-06
## study_hours  0.8306478  0.1577921  5.264191 1.408084e-07
```

- Given a student's study hours, we can now calculate the log-odds of passing the exam using the model equation.
- However, it would be easier to interpret the outcome in terms of probability rather than log-odds!

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 study\_hours)}}$$

- Given the coefficients estimates, can you calculate the probability of passing the exam given that the student studied for 6 hours?

Example - Logistic Regression Model

```
summary(model)$coefficients
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -3.3520313  0.7132017 -4.699977 2.601911e-06
## study_hours  0.8306478  0.1577921  5.264191 1.408084e-07
```

- Given a student's study hours, we can now calculate the log-odds of passing the exam using the model equation.
- However, it would be easier to interpret the outcome in terms of probability rather than log-odds!

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 study\_hours)}}$$

- Given the coefficients estimates, can you calculate the probability of passing the exam given that the student studied for 6 hours?

$$p = \frac{1}{1 + e^{-(-3.3520313 + 0.8306478 study\_hours)}} \approx 0.84$$

- We can also use R as the calculator.

```
# Coefficients provide the log-odds change for study_hours
coefficients <- coef(model)
beta_0 <- coefficients[1]
beta_1 <- coefficients[2]
```

```
# Assuming a student studied for 6 hours
log_odds <- as.numeric(beta_0 + beta_1 * 6)
probability <- 1 / (1 + exp(-log_odds))
print(probability)
```

```
## [1] 0.8364237
```

Classification Reminder

- We made a distinction earlier between regression (predicting a real value) and classification (predicting a discrete value).
- Within machine learning, logistic regression belongs to the family of supervised machine learning models. It is also considered a classification model, which means that it attempts to distinguish between classes (or categories).
- Recall that the goal for classification is to assign a class label to previously unseen data points as accurately as possible.
- Each object to be classified is represented as a pair (X, Y)
- Given a series of input/output pairs $\left(x^{(i)}, y^{(i)}\right)$:
  - For each observation $x^{(i)}$, we represent $x^{(i)}$ by a feature vector $[x_1, x_2, \ldots, x_n]$
  - We compute an output: a predicted class $\hat{y}^{(i)} \in \{0, 1\}$

How to do classification

- For each feature x_i, weight w_i tells us the importance of x_i.

$$z = \mathbf{w}^\top \mathbf{x} + b$$
$$y = \begin{cases} 1 & \text{if } z \geq r \\ 0 & \text{if } z < r \end{cases}$$

- Given multiple predictors x, we'll sum up all the weighted features and a bias term

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

- Then, the model will classify the object into one of the classes based on the threshold r.
- If the sum is high, we say y=1; if low, then y=0

But we want a probabilistic classifier

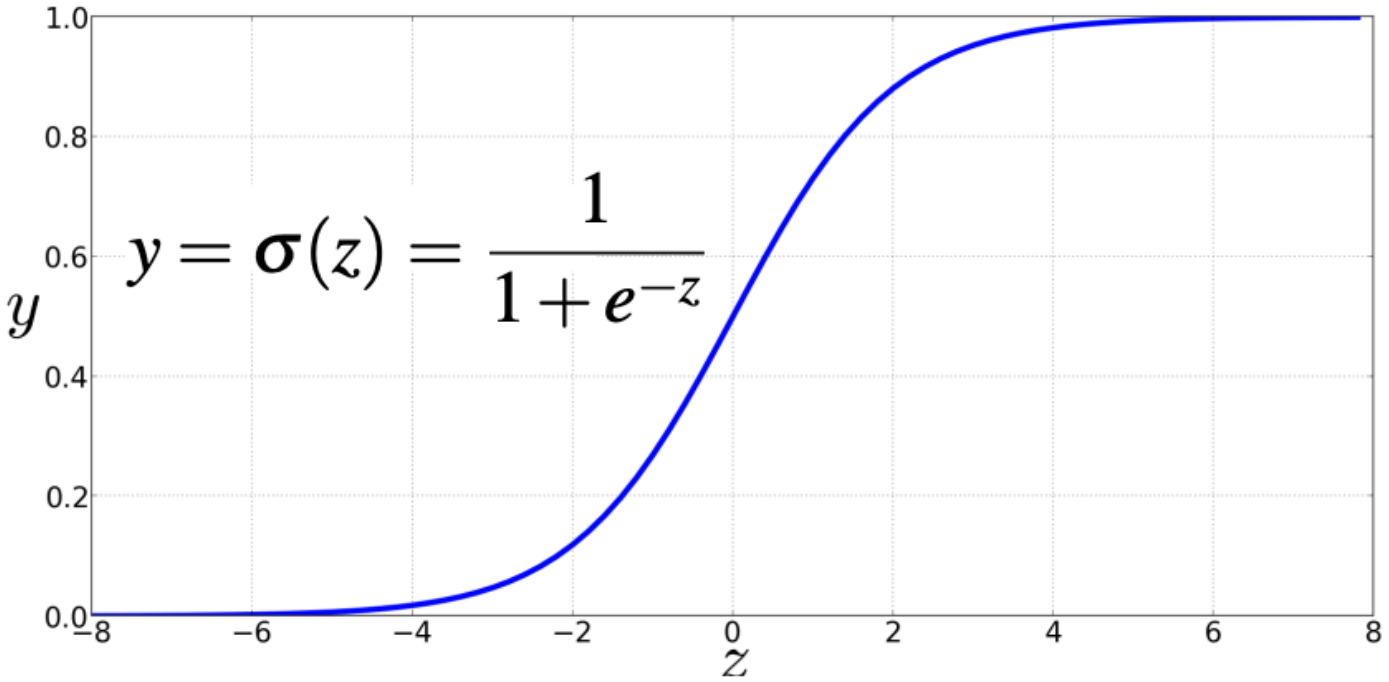
- We need to formalize "sum is high".
- We'd like a principled classifier that gives us a probability
- We want a model that can tell us:
  - $p(y = 1|x)$
  - $p(y = 0|x)$
- However, z isn't a probability, it's just a number!

$$z = \mathbf{w}^\top \mathbf{x} + b$$

- **Solution:** Remember how we use the sigmoid function to map the predicted values to probabilities that go from 0 to 1.

$$p = \sigma(z) = \frac{1}{1 + e^{-z}}$$

The very useful sigmoid or logistic function



Idea of logistic regression classification

- We'll compute $\mathbf{w}^\top \mathbf{x}$
- And then we'll pass it through the sigmoid function: $\sigma(\mathbf{w}^\top \mathbf{x})$
- And we'll just treat it as a probability
- In mathematical forms

$$
\begin{aligned}
P(y=1) &= \sigma(w \cdot x + b) \\
&= \frac{1}{1+e^{-(w \cdot x + b)}} \\
P(y=0) &= 1 - \sigma(w \cdot x + b)
\end{aligned}
$$

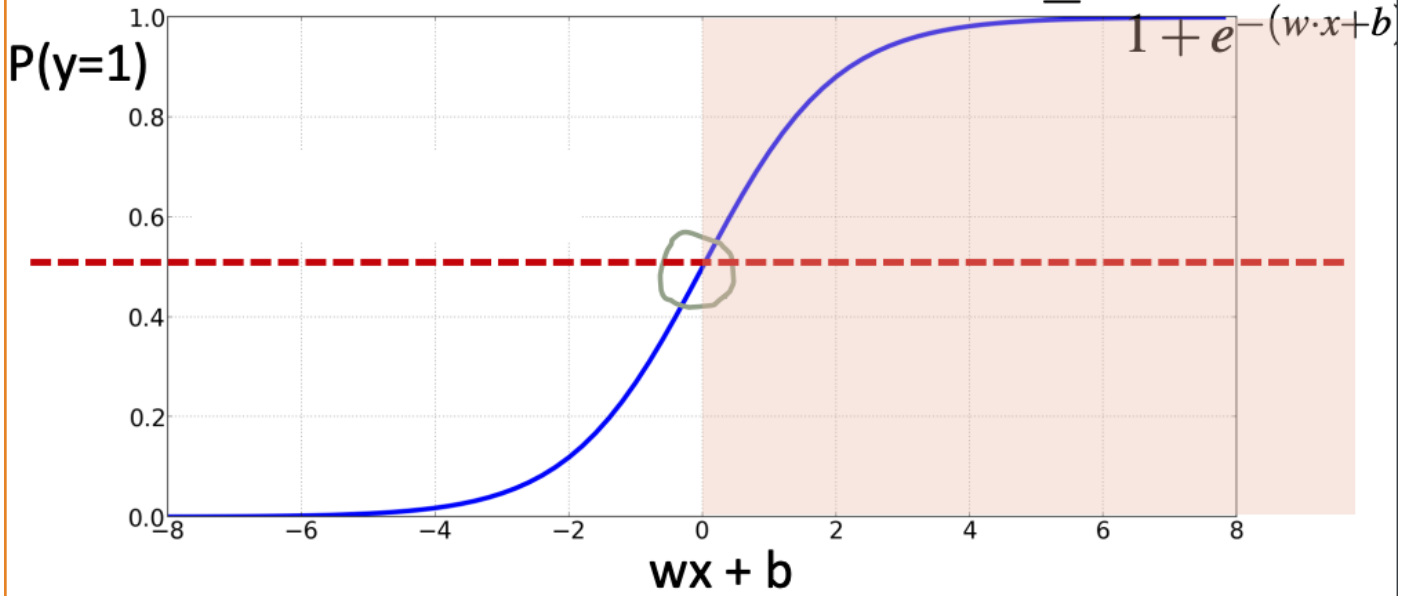
Idea of logistic regression classification

Then, to turn the probability into a classifier

$$\hat{y} = \begin{cases} 1 & \text{if } p(y=1|x) > 0.5 \\ 0 & \text{if } otherwise \end{cases}$$

Here, 0.5 is called the decision boundary.

The probabilistic classifier $P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+e^{-(w \cdot x + b)}}$



$$\hat{y} = \begin{cases} 1 & \text{if } p(y=1|x) > 0.5 \text{ if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Where did the w's come from

- Supervised classification:
  - We know the correct label y (either 0 or 1) for each x.
  - But what the system produces is an estimate, $\hat{y}$.
- We want to set w and b to minimize the distance between our estimate $\hat{y}$ and the true y.
- We need a distance estimator: a loss function or a cost function
  - A loss function: cross-entropy loss
  - An optimization algorithm: stochastic gradient descent
- An optimization algorithm to update w and b to minimize the loss (or cost).

Example

- Remember the scenario where we explored the relationship between body fat percentage and exercise level.
- Let's use logistic regression to create a classification boundary to predict the gender (male or female) based on body fat percentage and exercise hours per week.
  - We're dealing with binary classification (two classes).
  - Male and female are our class labels.

```
glimpse(data)
```

```
## Rows: 100
## Columns: 3
## $ exercise <dbl> 1.229156, 3.211000, 4.994624, 1.609496, 3.804740, 2.032657, 1…
## $ body_fat <dbl> 17.87578, 22.88305, 19.08977, 23.83017, 24.40467, 15.45556, 2…
## $ gender   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```

Example

- In machine learning, it's essential to have separate data for training our model and testing its predictions. This helps us evaluate the model's performance on unseen data and gives us an idea of how it might perform in real-world scenarios.
- Within the caret package in R, there's a convenient function called **createDataPartition()** that allows us to do just that:
  - 'data$gender': We specify which column to use for stratified sampling. In this case, we're using the gender column to ensure both training and testing sets have a similar distribution of genders.
  - 'p = .7': This indicates we want 70% of the data for training.

```
# Split the data into training and testing sets (70% train and 30% test)
trainIndex <- createDataPartition(data$gender, p = .7, list = FALSE)
data_train <- data[trainIndex,]
data_test  <- data[-trainIndex,]
```

Example

- Recall that in R, we can use the **glm()** function to fit different types of models.
- 'gender ~exercise + body_fat' is the formula. It means we're trying to predict gender based on exercise and body_fat features.
- Remember that we are using the training data to train the classifier.
- 'family = "binomial"' indicates that we are fitting a logistic regression model, suitable for binary classification tasks (male or female).

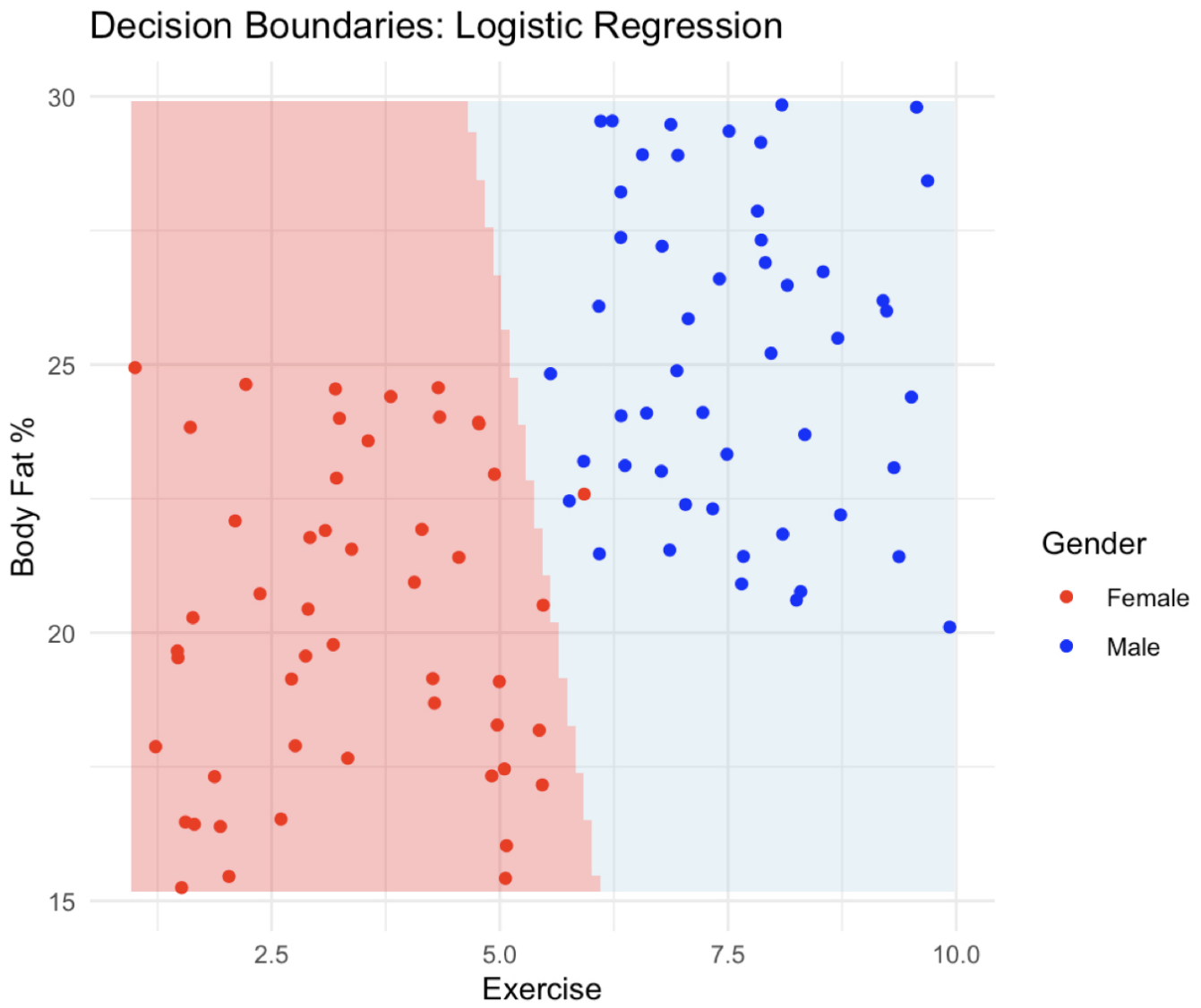
```
# Logistic Regression Model
logistic_model <- glm(gender ~ exercise + body_fat, data = data_train, family = "binomial")
```

Example

```
# Predict on the test set
probs <- predict(logistic_model, newdata = data_test, type = "response")
predictions <- ifelse(probs > 0.5, 1, 0)
```

- We are using the test data for model evaluation.
- The **'predict()'** function is used to get predictions from our logistic regression model. The 'type = "response"' argument ensures that we get probabilities as our output, which is a number between 0 and 1.
- Logistic regression outputs a probability that the given input point belongs to the '1' class. If the probability is greater than 0.5 (or 50%), we classify it as '1' (male in this dataset), otherwise, we classify it as '0' (female).
- This boundary is the set of points where the model is unsure (probability = 0.5) whether it should predict male or female.

Example

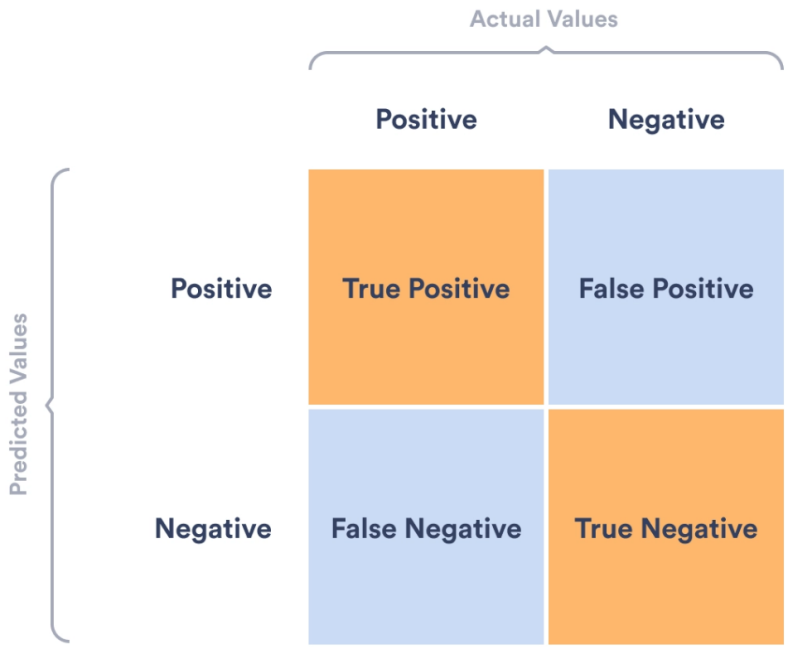

Decision Boundaries: Logistic Regression
Body Fat %
30
25
20
15
2.5
5.0
7.5
10.0
Exercise
Gender
Female
Male

Model Evaluation: Beyond Training

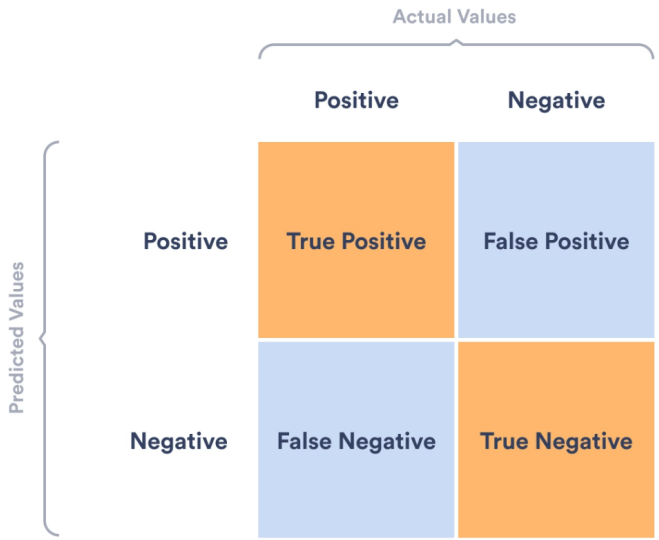
- In machine learning, the process doesn't end with just training the model.
- While we utilize a training set to construct our classifier, it's crucial to test its performance using a separate test set.
- This ensures that our model is being evaluated on data it hasn't seen before, truly testing its generalization ability.
- Some useful metrics include
  - Accuracy
  - Precision
  - Sensitivity (or Recall)
  - Specificity
  - F1 score

Confusion Matrix

- Before delving into performance metrics, it's essential to understand the foundation: the confusion matrix.
- The confusion matrix evaluates classification models, particularly when there are two or more potential outcomes. It consists of a table that displays four distinct outcomes based on actual versus predicted values.
- This matrix is pivotal for determining metrics like Recall, Precision, Specificity, Accuracy, and notably, the AUC-ROC curve.



Confusion Matrix



- **True Positives (TP):** The number of positive cases (e.g., the presence of a disease) that were correctly identified by the model.
- **False Positives (FP):** The number of negative cases that were mistakenly identified as positive.
  - This is also known as the Type 1 Error
- **False Negatives (FN):** The number of positive cases that were incorrectly identified as negative by the model
  - This is also known as the Type 2 Error
- **True Negatives (TN):** The number of negative cases that were correctly identified.

Evaluation Metrics

- **Accuracy**: Percentage of correct predictions. A good model should have high accuracy, but accuracy alone can be misleading, especially if classes are imbalanced.

$$\text{Accuracy} = \frac{TP + TN}{\#\text{ of predictions}}$$

- **Precision**: Of all positive predictions, how many are correct? A good model should minimize false positives to achieve high precision.

$$\text{Precision} = \frac{TP}{TP + FP}$$

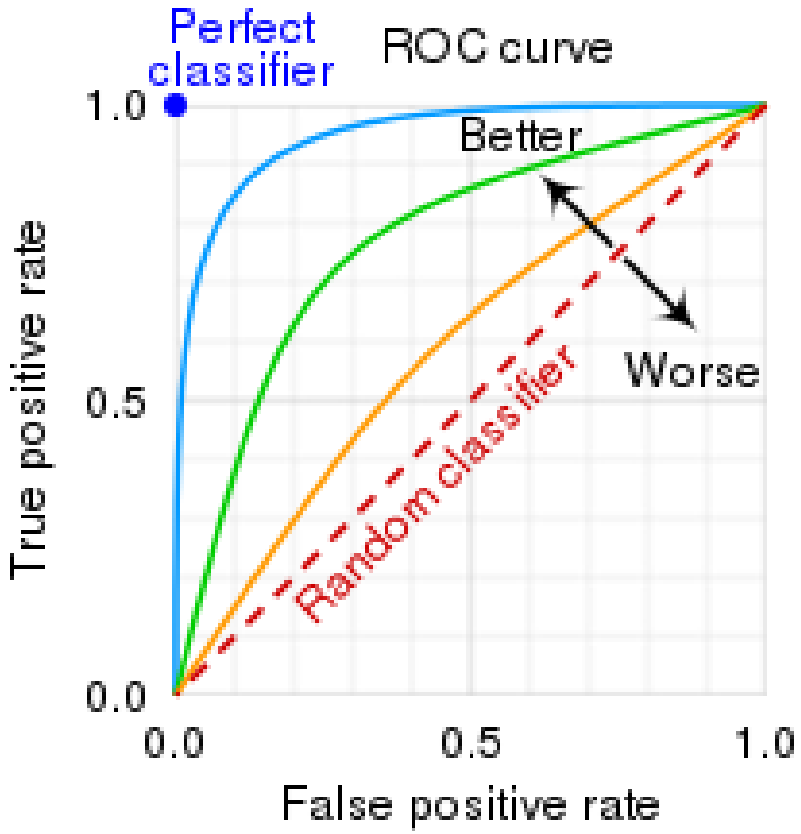
- **Recall (Sensitivity)**: Of all actual positive cases, how many were predicted as positive? A good model should capture as many positive cases as possible.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score**: It's a balance between Precision and Recall. A good model will have a high F1-score, indicating both high precision and high recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The ROC curve



ROC and AUC: ROC is the plot of True Positive Rate vs. False Positive Rate. AUC measures the area under the ROC curve.

Example

- Back to our example, how does our logistic classifier perform?
- The caret package in R provides the confusionMatrix function, which makes computing these metrics straightforward.

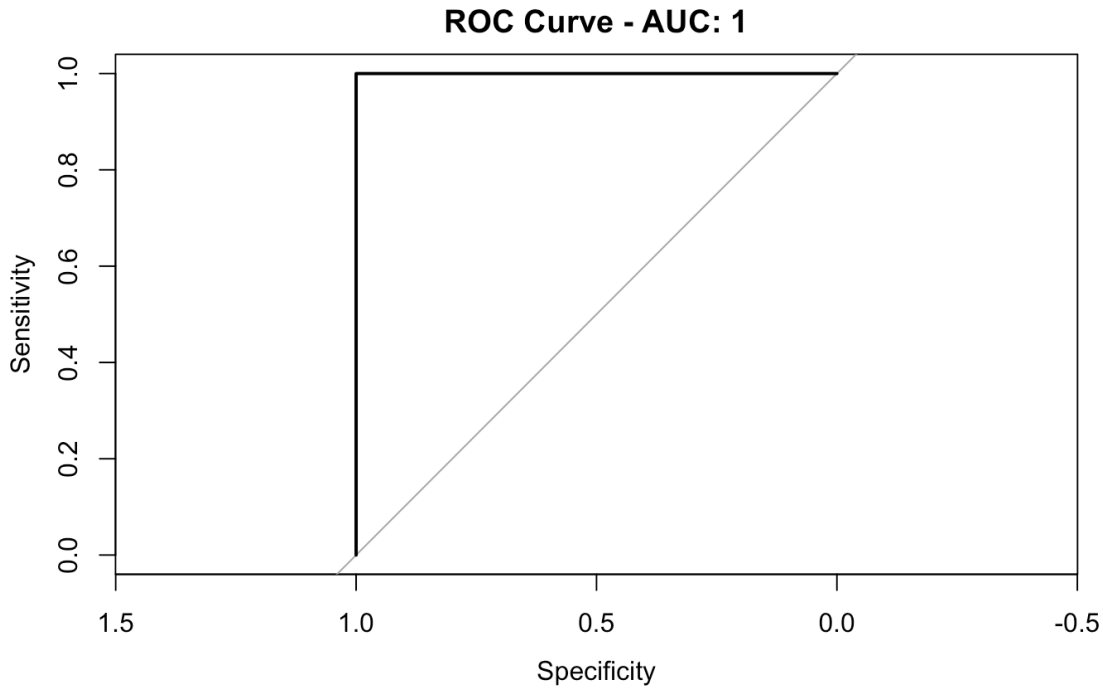
```
# Compute evaluation metrics
confusion <- confusionMatrix(as.factor(predictions), as.factor(data_test$gender))
accuracy <- confusion$overall['Accuracy']
precision <- confusion$byClass['Pos Pred Value']
recall <- confusion$byClass['Sensitivity']
F1 <- 2 * (precision * recall) / (precision + recall)
```

Example

```
[1] "Accuracy:  0.97"
[1] "Precision:  1"
[1] "Recall:  0.93"
[1] "F1 Score:  0.97"
```

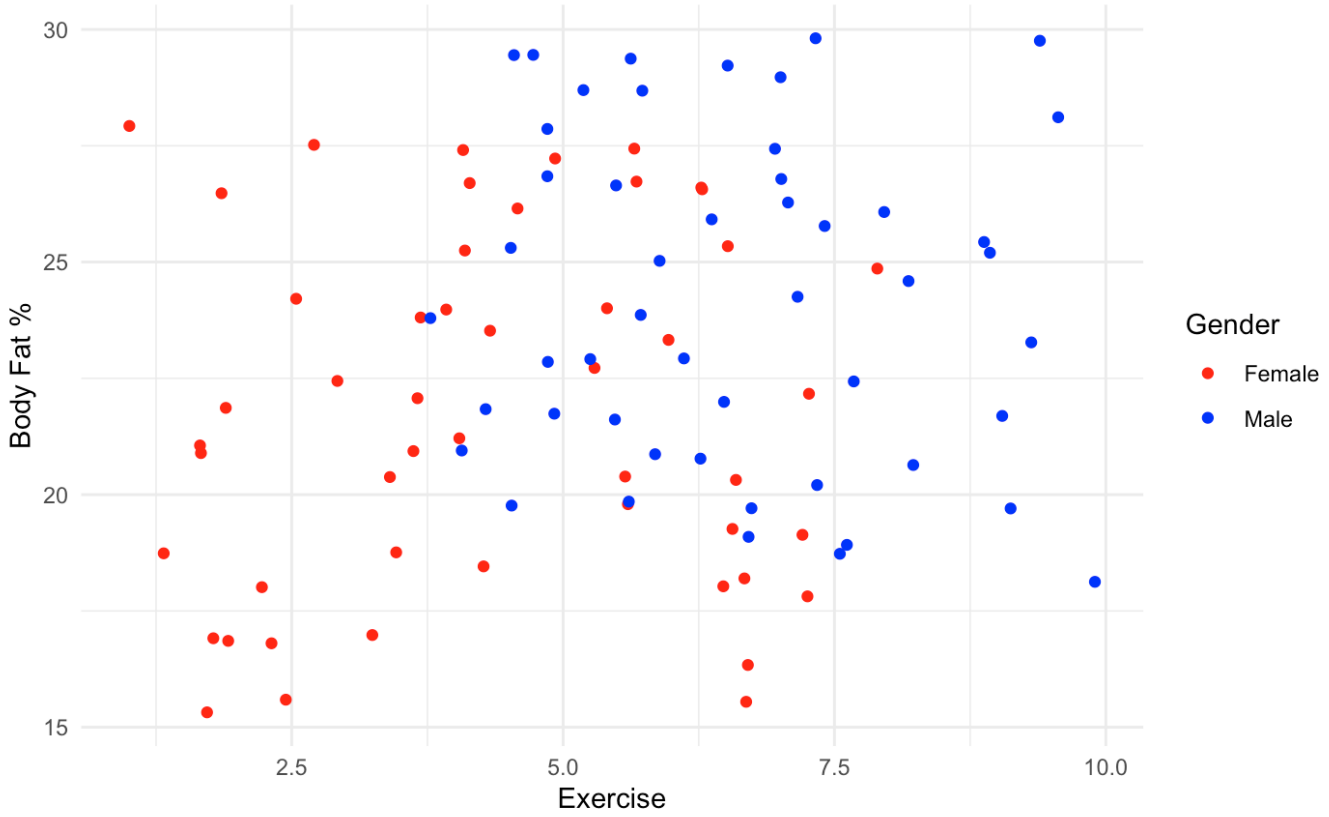
Our model is demonstrating high performance across all metrics. The 100% precision rate reveals that the classifier's positive predictions are entirely accurate. And there's only a 7% chance that the model might miss a positive case. An F1 score of 0.97 indicates a well-performing model in terms of both precision and recall.

Example - ROC and AUC

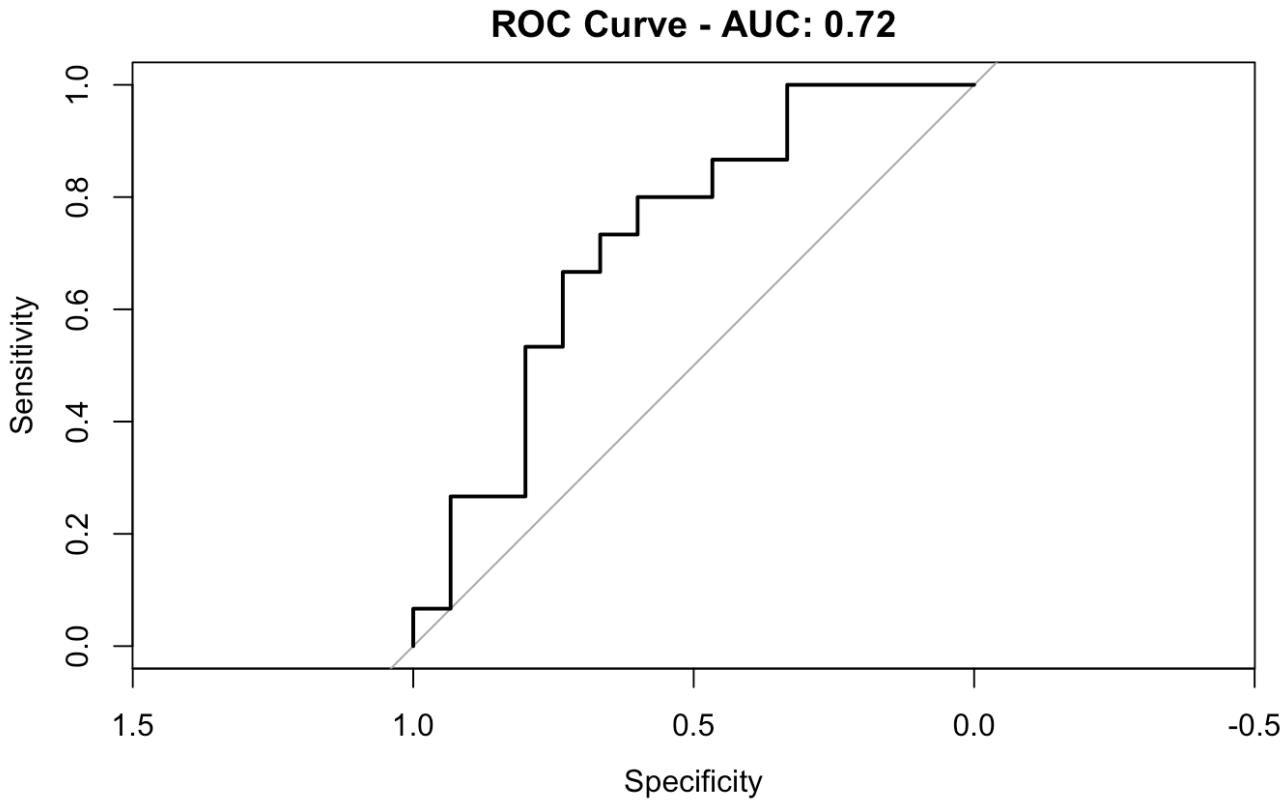


- The curve that reaches the top-left corner represents a near-perfect classifier. It indicates that for a large range of thresholds, the classifier has a high TPR (sensitivity) and a low FPR (1-specificity).
- If the AUC is 1 (or very close to 1), the classifier is able to distinguish between positive and negative cases with high accuracy across all thresholds.
- This might suggest that our data is linearly separable.

- But what if our data is not linearly separable?
- in real-world applications, it's rare to achieve a perfect classifier.



```
[1] "Accuracy:  0.63"
[1] "Precision:  0.7"
[1] "Recall:  0.47"
[1] "F1 Score:  0.56"
```



Today

Learning goals

- Understand the fundamental concepts of logistic regression.
- Comprehend the sigmoid function and its role in logistic regression.
- Implement logistic regression for binary classification using real-world examples.
- Evaluate the performance of a logistic regression classifier using appropriate metrics.
- Interpret and analyze ROC curves and AUC values.